Exact Voronoi diagram of smooth convex pseudo-circles: General predicates, and implementation for ellipses

Ioannis Z. Emiris[a], Elias P. Tsigaridas[b], George M. Tzoumas[c,*]

[a] *National and Kapodistrian University of Athens, Hellas*
[b] *INRIA Paris-Rocquencourt, UPMC, Univ. Paris 06, LIP6, France*
[c] *LE2I – University of Burgundy, France*

Abstract

We examine the problem of computing exactly the Voronoi diagram (via the dual Delaunay graph) of a set of, possibly intersecting, smooth convex pseudo-circles in the Euclidean plane, given in parametric form. Pseudo-circles are (convex) sites, every pair of which has at most two intersecting points. The Voronoi diagram is constructed incrementally. Our first contribution is to propose robust and efficient algorithms, under the exact computation paradigm, for all required predicates, thus generalizing earlier algorithms for non-intersecting ellipses. Second, we focus on InCircle, which is the hardest predicate, and express it by a simple sparse 5×5 polynomial system, which allows for an efficient implementation by means of successive Sylvester resultants and a new factorization lemma. The third contribution is our CGAL-based C++ software for the case of possibly intersecting ellipses, which is the first exact implementation for the problem. Our code spends about a minute to construct the Voronoi diagram of 200 ellipses, when few degeneracies occur. It is faster than the CGAL segment Voronoi diagram, when ellipses are approximated by k-gons for $k > 15$, and a state-of-the-art implementation of the Voronoi diagram of points, when each ellipse is approximated by more than 1250 points.

Keywords: Voronoi diagram, exact computation, CGAL implementation, InCircle predicate, parametric curve

1. Introduction

Computing the Voronoi diagram, and its dual Delaunay graph, of a set of sites in the plane has been studied extensively due to its links to other important questions, such as medial axis computations, but also its numerous

*Corresponding author
Email addresses: emiris@di.uoa.gr (Ioannis Z. Emiris), elias.tsigaridas@inria.fr (Elias P. Tsigaridas), George.Tzoumas@u-bourgogne.fr (George M. Tzoumas)

Figure 1: Left: Voronoi diagram and Delaunay graph of 26 ellipses; Right: Voronoi diagram of 7 intersecting ellipses

applications, including motion planning among obstacles, assembly, surface reconstruction, and crystallography (Anton, 2004). Our work is also motivated by other problems besides Voronoi diagrams: The predicates examined can be used to implement an algorithm for the convex hull of smooth convex pseudo-circles (Elber et al., 2001), whereas some of them appear in the computation of the visibility map among ellipses (Habert, 2005) (the software being developed for this problem uses our predicates). Recently, there has been some interest in applying our methods for contact detection in molecular dynamics(Ogarko and Luding, 2010)

Exact and certified geometric computing is an important enhancement to CAGD methods and an area of intense activity, since it exploits the modern power of geometric and algebraic algorithms which offer speed without jeopardizing robustness. Following this trend, our work guarantees the exactness of the dual Delaunay graph. This means that all combinatorial information is correct, namely the definition of the graph's edges, which, of course, have no geometric realization. The Voronoi diagram involves algebraic numbers for representing the vertices and bisector edges: our algorithms and software handle these numbers exactly. For instance, they can answer all point-location queries correctly. Hence, we say that our representation of the Voronoi diagram adheres to the principles of the exact computation paradigm, a distinctive feature in the realm of non-linear computational geometry. Our methods allow also for an approximation of the algebraic numbers with arbitrary precision, and this precision need not be fixed in advance, which is useful for drawing the diagram on the screen, or in cases where faster, yet approximate, computation is needed. We use the term *exact Voronoi diagram* to refer to a Voronoi diagram that is represented exactly via its dual Delaunay graph, while at the same time its vertices and edges can be approximated up to an arbitrarily high precision.

1.1. Previous work

Input sites have usually been linear objects, the hardest cases being line segments and polygons (Held, 2001; Karavelas, 2004); moreover, only the latter

(Karavelas, 2004) yields an exact output. The approximation of smooth curved objects by (non-smooth) linear or circular segments with no G^1 continuity may introduce artifacts and new branches in the Voronoi diagram, thus necessitating post-processing. It may even yield topologically incorrect results, as explained in (Ramamurthy and Farouki, 1999b).

The Voronoi diagram has been studied in the case of planar sites with curved boundaries (Ramamurthy and Farouki, 1999b; Alt et al., 2005), where topological properties are demonstrated, including the type of bisector curves, though the predicates and their implementation are not considered. There are works that compute the planar Voronoi diagram approximately: In (Hanniel et al., 2005), curve bisectors are traced within machine precision to compute a single Voronoi cell of a set of rational C^1-continuous parametric closed curves. The runtime of their implementation varies between a few seconds and a few minutes. It is briefly argued that the method extends to exact arithmetic, but without elaborating on the underlying algebraic computations or the handling of degeneracies. In (Ramanathan and Gurumoorthy, 2003), the boundary of the sites is traced with a prescribed precision, while Ramamurthy and Farouki (1999a) suggest working with lower-degree approximations of bisectors of curved sites. Finally, error-bounded bisector approximations of planar NURBS for the case of non-intersecting curves have been considered by Seong et al. (2008) and their implementation runtimes are within a couple of seconds for up to three sites. See also (Kim et al., 2001) for the Voronoi diagram of circles, which maintains topological consistency but not geometric exactness.

Curve-curve bisectors of parametric curves are considered in medial axis computations. In (Aichholzer et al., 2009) the medial axis of a simply connected planar domain is computed, using a divide and conquer algorithm. The boundary of the domain is approximated via G^1-continuous biarcs. The correct medial axis up to a predefined input accuracy is computed. The advantage of this type of approximation is that resulting medial axis is geometrically stable, exploiting the properties of G^1 continuity and the preservation of curvature maxima.

Few works have studied *exact* Voronoi diagrams for curved objects. In the case of circles, the exact and efficient implementation of Emiris and Karavelas (2006) is now part of (CGAL). There is also the very efficient and robust implementation of Held (2001), but relies on floating-point computations. For a more recent implementation that treats (non-intersecting) line segments and circular arcs, see (Held and Huber, 2009). Conics were considered in (Anton, 2004), but only in a purely theoretical framework. Moreover, the algebraic conditions derived were not optimal, leading to a prohibitively high algebraic complexity. In fact, the approach relied on eigenvector computations, hence was not exact.

Karavelas and Yvinec (2003b) have studied the properties of smooth convex, possibly intersecting, pseudo-circles in general position. They have shown that the Voronoi diagram of these sites belongs to the class of abstract Voronoi diagrams (Klein et al., 1993) and propose an incremental algorithm that relies on certain geometric predicates.

Our own previous work (Emiris et al., 2008) studied non-intersecting el-

lipses, and proposed exact symbolic algorithms for the predicates required by the incremental algorithm of Karavelas and Yvinec (2003b). Predicates SIDEOFBISECTOR, DISTANCEFROMBITANGENT were implemented. We established a tight bound of 184 complex tritangent circles to 3 ellipses (in fact, the result holds more generally for 3 conics). The upper bound on the number of real circles is still open, but we have examples with 76 real tritangent circles, when the ellipses intersect[1]. Predicate INCIRCLE had been implemented in MAPLE, and used a different polynomial system than the one in this paper. Some of its properties had been observed without proof; this is rectified below to yield an optimal method for resultant computation, using a more suitable polynomial system.

Emiris and Tzoumas (2008) proposed a certified method for INCIRCLE, relying on a Newton-like numerical subdivision, which exploited the geometry of the problem and exhibits quadratic convergence for non-intersecting ellipses, while for intersecting ones, the convergence is quadratic in general, except some special cases where the algorithm converges more slowly. The motivation behind this approach was to use a fast subdivision algorithm in order to filter out non-degenerate configurations and revert to slower algebraic algorithms only when necessary. The implementation made use of the double precision interval arithmetic library ALIAS[2]. In this work, we re-implement the proposed subdivision-based method in C++ using multi-precision floating point arithmetic. Experiments with state-of-the-art generic algebraic software on INCIRCLE (Emiris et al., 2008), and generic numeric solvers (Emiris and Tzoumas, 2008), imply that our specialized C++ implementation is much more efficient, both in the exact as well as in the certified numeric part.

1.2. Our contribution

In this article, which is an improved and finalized version of our work in Emiris et al. (2009), we extend previous results, that considered only non-intersecting ellipses, to smooth convex, possibly intersecting, pseudo-circles. This is the hardest step towards arbitrarily intersecting objects and requires reworking the predicates, especially INCIRCLE. At the same time, pseudo-circles are quite powerful. For instance, the Voronoi diagram in free (complementary) space of a set of arbitrarily intersecting convex objects coincides with the diagram of appropriate pseudo-circles (cf. Fig. 12). We propose algorithms for all necessary predicates and examine the algebraic operations required for an efficient exact implementation. The algebraic analysis of the required predicates, besides its theoretical importance, sheds light to the intrinsic complexity of computing with nonlinear objects in an exact way.

To express the Voronoi circle of parametric curves, we introduce a new 5×5 polynomial system, where we trade number of equations and unknowns for poly-

[1] G. Elber first showed us such examples. Recently, G. Moroz confirmed the same bound using adaptive sampling methods.

[2] http://www-sop.inria.fr/coprin/logiciels/ALIAS/

nomial degree. Although there is no determinant formula for this resultant, we exploit the structure of the system to find a succession of Sylvester determinants which eventually yields the resultant. Moreover, this approach bounds the degree of the algebraic numbers involved. In the case of arbitrary smooth parametric curves, Cor. 5 provides almost all the extraneous factors of the resultant. In the case of the conics we provide the complete factorization of the resultant (Th. 3). Our technique is quite general and could be used to other problems as well.

Finally, we present an implementation[3] in C++ for the case of ellipses that may intersect in pseudo-circles (cf. Fig. 1), extending CGAL's existing implementation for circles. It spends about 63 sec for the Voronoi diagram of 200 ellipses with few degeneracies. More importantly, it is faster than CGAL's segment and point Voronoi diagram, when ellipses are approximated by 16-gons and, respectively, by more than 1000 points roughly.[4] This is the first implementation under the exact computation paradigm for sites more complex than circles. Our benchmarks demonstrate that carefully implemented algebraic procedures incur a very reasonable overhead. Our ambition is to support arbitrary smooth convex objects and we hope CAGD shall benefit from the power of modern certified computing in the effort of reliable and robust software.

The rest of the paper is organized as follows. The next section introduces notation, and examines some basic operations on the sites, whereas section 3 studies the predicates of general parametric curves. Section 4 deals with InCircle, where we prove certain geometric and algebraic properties that lead to its efficient implementation. Section 5 presents our C++ implementation along with benchmarks measuring predicate performance, as well as how it competes against CGAL's Voronoi diagram for points and polygons, concluding with future work.

2. Preliminaries

Notation. Our input is smooth convex closed curves given in parametric form. Smoothness allows the tangent (and normal) line at any point of the curve to be well-defined. We denote by $C(t)$ a smooth closed convex curve parameterized by t. We refer to a point p on $C(t)$ with parameter value $\hat{t}$ by $p_{\hat{t}}$, or simply by $\hat{t}$, when it is clear from the context. By $\mathbf{C_t}^\circ$ we denote the region bounded by the curve $C(t)$. $\mathbf{C_t}$ is a smooth convex object (site), so that if p denotes a point in the plane, $p \in \mathbf{C_t} \iff p \in C(t) \cup \mathbf{C_t}^\circ$. When two sites intersect, we assume that their boundaries have at most two intersections, i.e., they form *pseudo-circles*. A curve $C(t)$ is given by the map

$$C(t) : [a, b] \ni t \mapsto (x(t), y(t)) = \left(\frac{X(t)}{W(t)}, \frac{Y(t)}{W(t)} \right), \tag{1}$$

[3] http://www.di.uoa.gr/∼geotz/vorell/

[4] This is for state-of-the-art implementations, as for simpler implementations about 300 points suffice to make them slower.

but actual denominators *can differ;* we use (1) for simplicity in our proofs. Here X, Y and W are polynomials in $\mathbb{Z}[t]$, with degrees bounded by d. Moreover, $a, b \in \mathbb{Q} \cup \{\pm\infty\}$. All algorithms, predicates and the corresponding analysis are valid for any parametric curve, even when the polynomials have different degrees, though we use (1) for simplicity. We assume that $W(t) \neq 0$, for any $t \in [a, b]$. The derivative of a polynomial $A(t)$ with respect to t is denoted by $A'(t)$. To simplify notation two sites defined by the closed curves $C_1(t)$ and $C_2(r)$ will be denoted by $\mathbf{C_t}$ and $\mathbf{C_r}$. We do the same for the defining curves, writing $C(t)$ and $C(r)$. When $d = 2$ the curves are conics; ellipses and circles are the only closed convex curves represented. Another example of a curve in the form of (1) is the *bean* curve, $t \mapsto (\frac{1+t^2}{t^4+t^2+1}, \frac{t(1+t^2)}{t^4+t^2+1})$.

We represent real algebraic numbers, i.e., the real roots of an integer polynomial, by the so-called *isolating interval* representation. For such a number α, this representation consists of a square-free polynomial, say A, which vanishes on α and an interval, say $[a, b]$, with rational endpoints that contains α and no other real root of A. The square-free part computation guarantees that the resulting polynomial does not have multiple roots. The polynomial may not be irreducible, since to derive an irreducible polynomial we need to perform factorization, which is a costly operation. In contrast with the computation of the square-free part which is cheap, as it consists of a GCD computation of a polynomial and its derivative.

Tangent and normal. Let $p_t(x(t), y(t))$ be a point on the curve $C(t)$. The equation of the tangent line at p_t is $(y - y(t))x'(t) - (x - x(t))y'(t) = 0$. After the substitution of the rational polynomial functions of $x(t)$ and $y(t)$, and multiplication of the equation by the denominators, we get a polynomial in $\mathbb{Z}[x, y, t]$, that is linear in x, y and of degree $\leq 2d - 2$ in t. The equation of the normal line at p_t is $(x - x(t))x'(t) + (y - y(t))y'(t) = 0$. As in the case of the tangent, after substitutions and elimination of the denominators we obtain a polynomial $N(x, y, t) \in \mathbb{Z}[x, y, t]$, which is linear in x, y and of degree $\leq 3d - 2$ in t.

Properties of pseudo-circles and relative position of sites. Now we characterize the relative position of sites $\mathbf{C_t}$, $\mathbf{C_r}$, i.e., whether they are separated, intersecting, externally or internally tangent, or if one is contained inside the other. The computation and characterization of all their bitangent lines suffices, due to the following properties: (i) By definition, C_t, C_r intersect as pseudo-circles iff they have at most two external bitangent lines. (ii) If one is contained inside the other, then there are 0 internal bitangent lines, and either 0 (boundaries separated) or up to a constant number (boundaries tangent) of external bitangent lines. If the number of external bitangent lines is greater than 2, then the pseudo-circles property is violated; otherwise, sites are internally tangent at 1 or 2 points. In this case, the pseudo-circles set is not in *general position.* We assume that our input is in general position, so that the properties of abstract Voronoi diagrams hold. Detection of internally tangent sites is performed by picking boundary points on either side of the tangency point, and testing whether they belong to the interior of the other site. (iii) If the sites do not

intersect, then they have 2 internal and 2 external bitangent lines (only one internal bitangent if they are externally tangent). (iv) Finally, if the sites admit more than 2 external bitangent lines, then they intersect in more than two points and then they do not form pseudo-circles.

The same technique decides the relative position of a point and a site since, if a point is interior to a site, there are no supporting lines tangent to the site. Consequently, this property is used to identify a site contained inside another, by considering a boundary point of one of the sites. In the case of conics (i.e. ellipses) the method of counting and characterizing the bitangents involves solving a polynomial of degree 4. There are cases where we need to compute the intersection points of sites $\mathbf{C_t}, \mathbf{C_r}$. These are solutions of the resultant of $X(t)W(r) - X(r)W(t)$ and $Y(t)W(r) - Y(r)W(t)$ with respect to t. This resultant is a polynomial in r of degree d^2.

3. Basic predicates

The insertion of a new site in the Voronoi diagram consists of the following: (i) Locate the nearest neighbor of the new site, (ii) Find a conflict between an edge of the current diagram and the new site, or detect that the latter is does not affect the Voronoi diagram (nor the Delaunay graph), as it is hidden by other sites (see Sec. 4.4 for a strict definition of hidden sites), (iii) Find the entire *conflict region,* defined as that part of the Voronoi diagram which changes due to the insertion of the new site, and update the dual Delaunay graph. The randomized complexity of an insertion (in a diagram with n sites) is $\mathcal{O}(\log^2 n)$ for disjoint sites and $\mathcal{O}(n)$ for intersecting or hidden sites. Computing an exact Voronoi diagram implies that we identify correctly all degenerate cases, including Voronoi circles tangent to more than 3 sites.

The basic predicates that are used in these steps are: SideOfBisector that performs nearest neighbor location; DistanceFromBitangent that determines if an infinite (unbounded) edge of the current Voronoi diagram will be modified, which can also be used to determine the existence of a Voronoi circle (Sec. 4.4); InCircle that determines if a vertex of the current Voronoi diagram is *in conflict* with the newly inserted site; EdgeConflictType which determines which part of an existing edge of the current Voronoi diagram will be modified due to the insertion of a new site. We briefly analyze predicates SideOfBisector and DistanceFromBitangent (abbreviated by SoB and DFB from now on). For a more elaborate analysis the reader may refer to (Emiris et al., 2009). Predicate InCircle is presented separately in the next section due to its higher complexity.

3.1. SideOfBisector

Given a site $\mathbf{C_t}$ and a point q in the plane, their (signed) *distance* equals $min_{x \in C(t)}||q - x||$ when $q \notin \mathbf{C_t}$ and $-min_{x \in C(t)}||q - x||$ when $q \in \mathbf{C_t}$, where $||\cdot||$ denotes the Euclidean norm. The absolute value of the distance equals the radius of the smallest circle centered at q tangent to $\mathbf{C_t}$. Given two sites

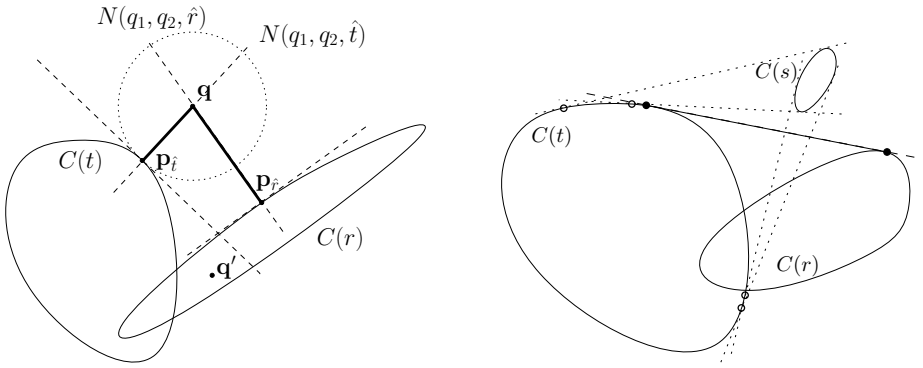

Figure 2: Left: Deciding SideOfBisector; Right: Deciding DistanceFromBitangent

$\mathbf{C_t}$ and $\mathbf{C_r}$ and a point $q = (q_1, q_2) \in \mathbb{Q}^2$, this predicate decides which site is closest to the point. If $q \notin \mathbf{C_t}$ and $q \in \mathbf{C_r}$, then q is closer to $\mathbf{C_r}$, due to the signed distance (cf. point q' in Fig. 2 left), otherwise, we compute the (squared) lengths of segments $qp_{\hat{t}}$ and $qp_{\hat{r}}$ as algebraic numbers and compare them. Let ζ denote the squared length of segment $q\,p_{\hat{t}}$, where $p_{\hat{t}} = (x(t), y(t))$, for $t = \hat{t}$. Then we have that $\zeta - (x(t) - q_1)^2 - (y(t) - q_2)^2 = 0$. Recall (from Sec. 2) the equation of the normal line passing through q satisfying $N(q_1, q_2, t) = 0$. We can eliminate t from these two equations by computing $R(\zeta) = \mathsf{Res}_t(\zeta - (x(t) - q_1)^2 - (y(t) - q_2)^2, N(q_1, q_2, t))$ (i.e. the resultant of the two polynomials with respect to t). This yields a polynomial $R \in \mathbb{Z}[\zeta]$, of degree $3d - 2$, the smallest positive real root of which is equal to $\|q - p_{\hat{t}}\|^2$.

The degree of the real algebraic numbers involved in the predicate is $3d - 2$. For conics, this degree becomes 4, and is optimal, as shown in (Emiris et al., 2008), where it was obtained using the pencil of two ellipses. Note that we do not compute the bisector of the two curves. The parametric bisector of two rational parametric curves is a bivariate polynomial of degree $4d - 2$ in each variable, while in the cartesian space (as a polynomial in x, y) the degree is even higher (see Elber and Kim (1998) and Emiris et al. (2008) for the case of ellipses). Therefore, the approach presented here is preferred, as we are dealing with algebraic numbers of lower degree.

3.2. DistanceFromBitangent

Consider two sites, $\mathbf{C_t}$ and $\mathbf{C_r}$, and their bitangent line, which leaves both sites on the right-hand side, as we move from the tangency point of $C(t)$ to the tangency point of $C(r)$; such a bitangent appears in Fig. 2 right and is called a CCW bitangent line, because it can be considered as a circle of infinite radius oriented in CCW direction. This line divides the plane into two halfplanes and DFB decides whether a third site, $\mathbf{C_s}$, lies in the same halfplane as the other two. The realization of this predicate consists in deciding the relative position of $\mathbf{C_s}$ with respect to the bitangent line. We define the outcome of the predicate to be negative, that is $\mathrm{DFB}(\mathbf{C_t}, \mathbf{C_r}, \mathbf{C_s}) < 0$, iff $\mathbf{C_s}$ does not lie in the same

halfplane as $\mathbf{C_t}$ and $\mathbf{C_r}$ with respect to the CCW bitangent line. We split the problem to two sub-problems: compute the external bitangent of interest, and decide the relative position of the third site with respect to this bitangent.

To compute all bitangents of $C(t), C(r)$, we consider the equation of a tangent line to $C(t)$ that also crosses $C(r)$. For the line to be tangent to both sites, the discriminant of the corresponding polynomial should vanish. Among the real roots of the discriminant are the values of the parameter that correspond to the tangency points, which allow us to compute the implicit equations of the bitangent lines. We characterize the bitangents as external or internal by computing their relative position with respect to (rational) points inside the sites.

To decide the position of $\mathbf{C_s}$ with respect to the CCW bitangent line, we first check if the line crosses $\mathbf{C_s}$. If this is the case, then the predicate is answered immediately, since $\mathbf{C_s}$ cannot lie within the same halfplane as $\mathbf{C_t}$ and $\mathbf{C_r}$. If this line does not cross $\mathbf{C_s}$, then to decide in which halfplane $\mathbf{C_s}$ lies, it suffices to compute the sign of the evaluation of the polynomials of the bitangent over an interior (rational) point of $\mathbf{C_s}$. To check whether the bitangent line crosses $\mathbf{C_s}$, we consider the tangency points of all the bitangents of $C(t)$ and $C(s)$, shown with circular marks in Fig. 2 right. We then decide the position of $\mathbf{C_s}$ by the ordering of the aforementioned points *and* the tangency point of the bitangent and $C(t)$, shown with solid circular mark in the same figure. The algebraic numbers considered in this predicate are of degree $\leq 4(d-1)^2$.

The predicates examined so far, apart from the computation of the Voronoi diagram, can also be used for the computation of the convex hull of smooth convex pseudo-circles.

4. InCircle

This section introduces a polynomial system for expressing the Voronoi circle, leading to a robust and fast implementation. Each point q on a Voronoi edge that is equidistant to two sites $\mathbf{C_t}$ and $\mathbf{C_r}$ is the center of a disk tangent to the boundaries $C(t)$ and $C(r)$ of the sites. This disk is called an *internal bitangent Voronoi disk*, if it is contained in $\mathbf{C_t} \cap \mathbf{C_r}$ and an *external bitangent Voronoi disk*, if it lies in the complement of ${\mathbf{C_t}}^\circ \cup {\mathbf{C_r}}^\circ$. Similarly, a Voronoi vertex equidistant from three sites $\mathbf{C_t}$, $\mathbf{C_r}$ and $\mathbf{C_s}$, is the center of a disk tangent to the boundaries of $\mathbf{C_t}$, $\mathbf{C_r}$ and $\mathbf{C_s}$. Such a disk is called an *internal tritangent Voronoi disk*, if it is contained in $\mathbf{C_t} \cap \mathbf{C_r} \cap \mathbf{C_s}$ and an *external tritangent Voronoi disk* , if it lies in the complement of ${\mathbf{C_t}}^\circ \cup {\mathbf{C_r}}^\circ \cup {\mathbf{C_s}}^\circ$.

Given sites $\mathbf{C_t}$, $\mathbf{C_r}$, $\mathbf{C_s}$ in this order, we denote their associated Voronoi disk by V_{trs} iff their tangency points on the disk are in CCW direction. In this case, V_{trs} is a CCW Voronoi disk, and V_{tsr} is a CW Voronoi disk. Since the Voronoi diagram of smooth convex pseudo-circles is an abstract Voronoi diagram, given 3 sites, there may exist at most one CCW Voronoi disk and at most one CW Voronoi disk (cf. Fig. 6, where sites $\mathbf{C_t}$,$\mathbf{C_r}$,$\mathbf{C_s}$ correspond to ellipses 1,2 and 3 resp.). Note that the boundary of the Voronoi disk is the Voronoi circle.

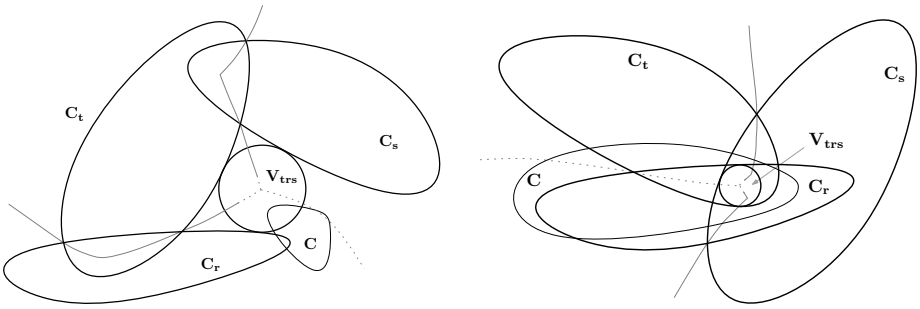

Figure 3: Conflict of site $\mathbf{C}$ with the external Voronoi disk (left) or with the internal one (right) of $\mathbf{C_t}$, $\mathbf{C_r}$ and $\mathbf{C_s}$. Solid line: Voronoi edges. Dotted line: Subset of the Voronoi diagram that is in conflict with $\mathbf{C}$

Point q on the Voronoi diagram and site $\mathbf{C}$ are in *conflict* if the Voronoi disk associated with q is an internal Voronoi disk contained in $\mathbf{C}^\circ$, or an external Voronoi disk intersecting $\mathbf{C}^\circ$ (cf. Fig. 3). Given sites $\mathbf{C_t}$, $\mathbf{C_r}$, $\mathbf{C_s}$, let V_{trs} be their Voronoi disk and $\mathbf{C}$ be a query site. If V_{trs} is an external Voronoi disk, we say that $\mathbf{C}$ is in *conflict* with V_{trs} if V_{trs} is intersecting $\mathbf{C}^\circ$. If V_{trs} is an internal Voronoi disk, we say that $\mathbf{C}$ is in *conflict* with V_{trs} if V_{trs} is included in $\mathbf{C}^\circ$. This type of conflict is called *vertex conflict*, because when a newly inserted site $\mathbf{C}$ is in conflict with a Voronoi disk, the center of that disk corresponds to a Voronoi vertex that will no longer exist in the new Voronoi diagram, as shown in Fig. 3.

InCircle decides if a newly inserted site $\mathbf{C}$ is in *conflict* with V_{trs}. A degeneracy arises when $\mathbf{C}$ is also tangent to V_{trs}. Given that V_{trs} exists, the predicate is computed as follows: (i) Solve the algebraic system that expresses the Voronoi circle. Among the solutions (which correspond to various tritangent circles, cf. Fig. 4 right), find V_{trs}. (ii) Determine the relative position of $\mathbf{C}$ with respect to V_{trs}. Each step is explained in the subsections that follow.

4.1. Computing the Voronoi circle

The polynomial system expressing all circles tangent to $\mathbf{C_t}$, $\mathbf{C_r}$, $\mathbf{C_s}$ is:

$$N(x,y,t) = N(x,y,r) = N(x,y,s) = M_{tr}(x,y,t,r) = M_{ts}(x,y,t,s) = 0 \quad (2)$$

and has a mixed volume of 512 for the case of conics, which immediately provides an upper bound on the number of roots in $\mathbb{C}^*$. The first 3 equations correspond to normals at points t, r, s on the 3 given sites. All normals go through the Voronoi vertex (x, y). The last two equations are linear in x, y and force (x, y) to be equidistant from the sites: each one corresponds to the (perpendicular) bisector of the segment between two footpoints (cf. Fig. 4 left). This system was also used by Ramamurthy and Farouki (1999a). Elimination of x, y from $M_{tr}(x, y, t, r), N(x, y, t), N(x, y, r)$ yields the bisector of two sites with respect to t, r. System (2) shall be solved over $\mathbb{R}$, thus yielding a set of solution vectors in $\mathbb{R}^5$. Only one solution vector corresponds to V_{trs}. There exist solution vectors

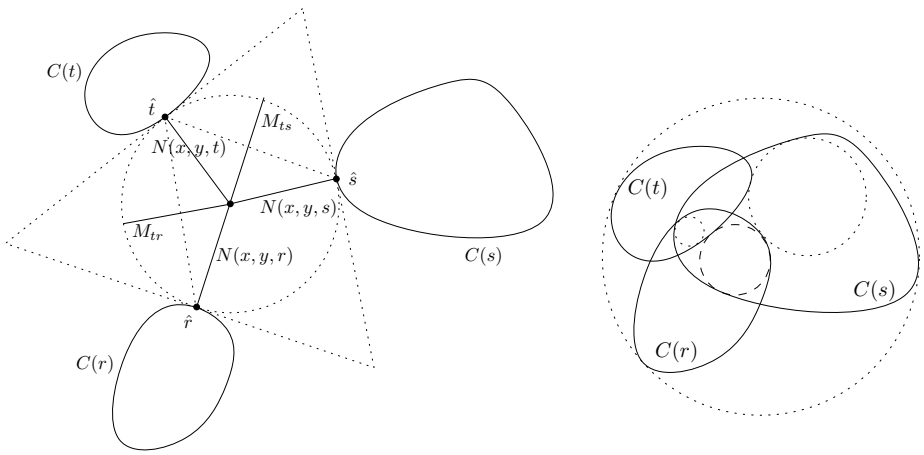

Figure 4: Left: An external tritangent circle; Right: Various tritangent circles. Dotted line: CCW(t,r,s), Dashed line: CW(t,r,s)

with CCW orientation of the tangency points, and solution vectors with CW orientation which do not correspond to the Voronoi circle we are looking for. They just correspond to some tritangent circle (cf. Fig. 4 right).

The *resultant* of $n+1$ polynomials in n variables is an irreducible polynomial in the coefficients of the polynomials which vanishes iff the system has a complex solution. Irreducibility occurs for generic coefficients; otherwise, resultants can be factorized. In particular, sparse (or toric) resultants express the existence of solutions in $(\mathbb{C}^*)^n$ (Cox et al., 2005). We employ vectors $\alpha = (\alpha_1, \ldots, \alpha_n) \in \mathbb{N}^n$, where $|\alpha|$ is the 1-norm and we write x^α for $\Pi_i x_i^{\alpha_i}$.

Proposition 1. *We are given polynomials $F_0, \ldots, F_n \in \mathbb{K}[x_1, \ldots, x_n]$ over a field $\mathbb{K}$, such that $F_i = \sum_{0 \le |\alpha| \le d_i} u_{i,\alpha} x^\alpha$ is of total degree d_i, where $0 \le i \le n$. Their resultant with respect to $x_1, \ldots, x_n$ is homogeneous of degree $d_0 \cdots d_{j-1} d_{j+1} \cdots d_n$ in $u_{j,\alpha}$, where $0 \le |\alpha| \le d_j$, and $j \in [0, n]$. This means, for any $\lambda \in \mathbb{K}$, that $\mathsf{Res}(F_0, \ldots, \lambda F_j, \ldots, F_n) = \lambda^{d_0 \cdots d_{j-1} d_{j+1} \cdots d_n}\ \mathsf{Res}(F_0, \cdots, F_n)$. The total degree of the resultant is $\sum_{j=0}^{n} d_0 \cdots d_{j-1} d_{j+1} \cdots d_n$.*

Lemma 2. *Let $f(x) = T^2 a_n x^n + T a_{n-1} x^{n-1} + \sum_{i=0}^{n-2} a_i x^i$ and $g(x) = T^2 b_n x^n + T b_{n-1} x^{n-1} + \sum_{i=0}^{n-2} b_i x^i$. Then the resultant of f and g with respect to x is a multiple of T^4.*

Proof. Let S be the $2n \times 2n$ Sylvester matrix of f, g with respect to x. Their resultant equals $\det S$. We shall exploit its properties (for a more elaborate proof, with matrices shown see (Tzoumas, 2009, p. 89)). Denote by S_i the new matrix S after step i: Divide the first two columns by T^2 and T respectively and get $|S| = T^3 |S_1|$. Then, divide rows 2 and $n+2$ by T and rows i and $n+i$, $3 \le i \le n$ by T^2 and get $|S| = T^5 T^{2(n-2)2} |S_2|$. Now, multiply column 4 by T and columns 5 to $2n$ by T^2 to remove denominators, thus obtaining $|S| = T^5 T^{2(2n-4)} T^{-1} T^{(-2)(2n-4)} |S_3| \iff |S| = T^4 |S_3|$. □

It is impossible to compute the resultant of 5 arbitrary polynomials as a determinant, so we apply successive Sylvester determinants, i.e., optimal resultant formulae for $n = 1$. This typically produces extraneous factors but, by exploiting the fact that some polynomials are linear, and that none contains all variables, we shall provide the *complete* factorization of the computed polynomial; we focus on conics for simplicity, but our approach holds for any parametric curve. We denote by $\Pi(t)$ the resultant of (2) when eliminating all variables except t: it is, generally, an irreducible univariate polynomial and vanishes at the values of t that correspond to the complex tritangent circles. Recall that the curves are defined by (1).

Theorem 3. *If* $\Pi(t)$ *is the resultant of (2) as above, then* $\mathsf{Res}_{xy}(R_1, R_2, N(x, y, t)) = \Pi(t)W(t)^{40}(Y(t)W'(t)-Y'(t)W(t))^{36}$*, where,* $R_1 = \mathsf{Res}_r(M_{tr}(x, y, t, r), N(x, y, r))$*,* $R_2 = \mathsf{Res}_s(M_{ts}(x, y, t, s), N(x, y, s))$ *and the degree of* Π *is 184.*

Proof. All polynomials belong to $\mathbb{Z}[x, y, t, r, s]$: we shall eliminate x, y, r, s to obtain the univariate resultant in $\mathbb{Z}[t]$. Polynomials $N(x, y, t)$, $N(x, y, r)$, and $N(x, y, s)$ are of total degree 5, linear in x, y and of degree 4 in the parameter. Polynomials $M_{tr}(x, y, t, r)$ and $M_{ts}(x, y, t, s)$ are of total degree 9, linear in x and y and of degree 4 in the parameters. First, we eliminate r from $M_{tr}(x, y, t, r)$ and $N(x, y, r)$: $R_1(x, y, t) = \mathsf{Res}_r(M_{tr}(x, y, t, r), N(x, y, r))$. It is of total degree 22, of degree 6 in x, in y, and in x, y, and 16 in t. We do the same for $M_{ts}(x, y, t, s), N(x, y, s)$ and obtain: $R_2(x, y, t) = \mathsf{Res}_s(M_{ts}(x, y, t, s), N(x, y, s))$, which follows the same degree pattern. It remains to compute the final polynomial in $\mathbb{Z}[t]$: $R_3(t) = \mathsf{Res}_{x,y}(R_1(x, y, t), R_2(x, y, t), N(x, y, t))$.

Let $N(x, y, t) = D(t)y + A(t)x + C(t)$, where $D(t) = W(t)(Y(t)W'(t) - Y'(t)W(t))$ of degree 4. To compute R_3, we solve $N(x, y, t)$ for y and substitute in R_1, R_2. This introduces denominators, eliminated by multiplying by $D(t)^6$. Then, we take the Sylvester resultant with respect to x (denominator $D(t)$ vanishes if $N(x, y, t)$ is vertical; in this case, we compute R_3 by solving $N(x, y, t)$ for x and proceed symmetrically with the resultant with respect to y). We apply prop. 1 as follows:

$$\begin{aligned}\mathsf{Res}_{xy}(D^6\,R_1, D^6\,R_2, y + \tfrac{Ax+C}{D}) = D^{36}D^{36}\mathsf{Res}_{xy}(R_1, R_2, y + \tfrac{Ax+C}{D}) = \\ = D^{36}\mathsf{Res}_{xy}(R_1, R_2, D(y + \tfrac{Ax+C}{D})) = D^{36}R_3.\end{aligned} \tag{3}$$

By prop. 1, the degree of the resultant in t is $16(6\cdot1)+16(6\cdot1)+4(6\cdot6) = 336$. The previous discussion implies that $(W(t)(Y(t)W'(t)-Y'(t)W(t)))^{36}$, the leading coefficient of y in $N(x, y, t)$, appears as an extra factor. After substitution of y in R_1 and R_2, we obtain polynomials of the form $W(t)^2c_6x^6 + W(t)c_5x^5 + c_4x^4 + c_3x^3 + \cdots + c_0$. Lemma 2 implies the resultant of two such polynomials contains $W(t)^4$ as a factor. Therefore, the degree of Π is $336-4\cdot36-2\cdot4 = 184$. □

This theorem provides an upper bound of 184 complex tritangent circles to 3 conics. Numeric examples show that the bound is tight. This bound was also obtained in (Emiris et al., 2008) using a different system. Now, we describe the factorization for arbitrary degree curves based on prop. 1.

Corollary 4. *We are given $R_0, R_1, R_2 \in \mathbb{K}[x, y]$, where the total degree of R_1 and R_2 is n in x, in y, and in x and y together, and $R_0 = Dy + Ax + C$, where $AD \neq 0$, then* $\mathsf{Res}_x(\mathsf{Res}_y(R_0, R_1), \mathsf{Res}_y(R_0, R_2)) = D^{n^2}\mathsf{Res}_{xy}(R_0, R_1, R_2)$.

Corollary 5. *The degree of the resultant of (2) for general parametric curves, as in (1), is bounded by $(3d-2)(5d-2)(9d-2)$, after dividing out the factor of $(W(t)(Y(t)W'(t) - Y'(t)W(t)))^{(5d-2)^2}$.*

A more careful analysis may exploit term cancellations to yield a tighter bound.

If we solve the resultant of system (2), we obtain one coordinate of the solution vectors (in isolating interval representation). There are methods to obtain the other variables, too. For instance, plugging a value of t in the bisector of $\mathbf{C_t}$ and $\mathbf{C_r}$ allows us to find r (cf. Sec. 4.2). The resultant allows us to detect *degenerate* configurations: 4 sites tangent to the same Voronoi disk. Consider triplets $\mathbf{C_t}$, $\mathbf{C_r}$, $\mathbf{C_s}$ and $\mathbf{C_t}$, $\mathbf{C_r}$, $\mathbf{C_h}$. Let $\Pi_1(t), \Pi_2(t)$ be the resultants of (2) respectively. If the triplets admit an identical Voronoi circle, then $gcd(\Pi_1, \Pi_2) \neq 1$. If $gcd(\Pi_1, \Pi_2) \neq 1$, the triplets *may* have an identical solution vector, which is verified by looking at the other coordinates analogously to (Emiris et al., 2008).

4.2. Choosing the proper solution

We consider the question of choosing, among all solutions of the polynomial system corresponding to the tangency points $(\hat{t}, \hat{r}, \hat{s})$ of a tritangent circle, the one that corresponds to Voronoi circle V_{trs}. To eliminate irrelevant solutions, consider tangency points $p_{\hat{t}}$, $p_{\hat{r}}$, $p_{\hat{s}}$ for a solution triplet $\hat{t}, \hat{r}, \hat{s}$. The tangency points corresponding to V_{trs} satisfy CCW($p_{\hat{t}}$, $p_{\hat{r}}$, $p_{\hat{s}}$) and we disregard the rest of the solution vectors.

Now we distinguish an external and an internal tritangent circle from the rest of the tritangent circles. The tangency points define the former iff the tangent line of the Voronoi circle at each tangency point separates its adjacent site from the other two tangency points, see Fig. 4 left. Even if the tangent line intersects the other sites (not shown in the example), the tangency points are still separated. Checking that the tangency points correspond to an internal circle is more complex, because an argument symmetric to the external case (i.e. the internal tritangent circle is such that all three tangency points are on the same side of the tangent line as the site, and inside the site) does not apply, due to the fact that the internal tritangent circle may be "locally" inside the curve, but not "globally"). Determining whether the tangency points correspond to an internal circle can be performed by applying the following lemma:

Lemma 6 (Emiris et al. (2009)). *Given intersecting sites $\mathbf{C_t}$ and $\mathbf{C_r}$, consider their bitangent circle B_{tr} at points $\hat{t}$ and $\hat{r}$ respectively with $\hat{t}, \hat{r} \in \mathbf{C_t} \cap \mathbf{C_r}$. Then B_{tr} is an internal bitangent circle iff B_{tr} has the smallest radius among all bitangent circles of $C(t)$ and $C(r)$ tangent at $\hat{t}$, and the radius of B_{tr} is bounded by the radius of curvature of $C(t)$ at $\hat{t}$ and the radius of the self-bitangent circle of $C(t)$ at $\hat{t}$.*

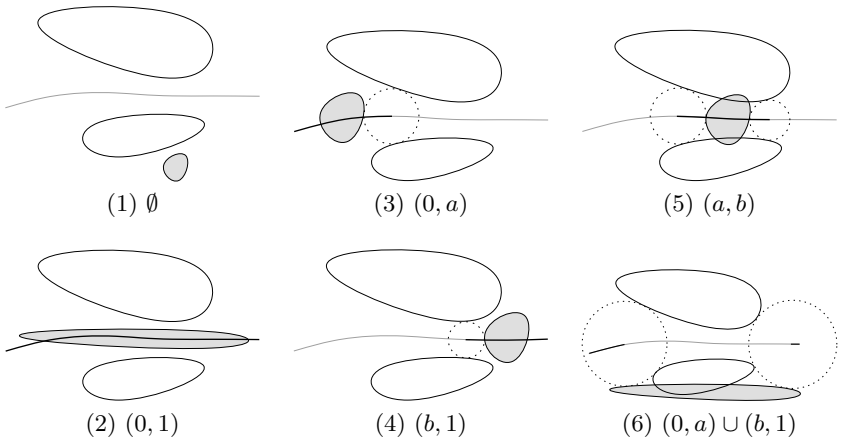

Figure 5: The six cases for the shadow region of site **C** (shaded site) with respect to the bisector (gray line) of two other sites. Voronoi circles are shown in dotted line

Finally, an external bitangent circle B_{tr} can be computed as in (Emiris et al., 2008), by considering the tangent line at $\hat{t}$, and the boundary of the convex hull of $\hat{t} \cup C_r$.

In our implementation, the above properties (computing internal and external bitangent circles) are used through an improved version of the subdivision scheme of (Emiris and Tzoumas, 2008) that converges to the tangency point of the Voronoi circle. We have incorporated Lem. 6 so that we are able to handle internal Voronoi circles as well. This way, not only are we able to choose the proper solution of (2), we can also approximate the root before computing the resultant. Therefore the resultant is computed only when our approximation cannot guarantee correctness, i.e., when the precision chosen is not enough, or when we are in a degenerate case (cf. Sec. 5).

4.3. Deciding conflict

Since the Voronoi circle is expressed algebraically, it is not clear how to decide its relative position with respect to the query site, i.e., how to apply the method of Sec. 2. There is an interesting geometric property that allows us to determine a conflict by comparing tangency points on the boundary of one site. These points have "easy" representation as roots of the resultant of system (2).

Definition 7. *The* shadow region $S_{tr}(\mathbf{C})$ *of a site* $\mathbf{C}$ *with respect to the bisector* $B(t, r)$ *of two sites* $\mathbf{C_t}$ *and* $\mathbf{C_r}$ *is the locus of points* q *of* $B(t, r)$ *that are in conflict with* $\mathbf{C}$*.*

Alternatively, the shadow region can be seen as the part of the Voronoi diagram of two sites that will disappear (i.e., the part of the bisector that will be invalidated) due to the insertion of a third site. An important property of abstract Voronoi diagrams is that the bisector of two sites is homeomorphic to the open

interval (0,1) and the shadow region has one out of six possible forms: $\emptyset$, $(0,1)$, $(0,a)$, $(b,1)$, (a,b) and $(0,a)\cup(b,1)$, where $a,b\in(0,1)$ (Karavelas and Yvinec, 2003b; Klein et al., 1993), as shown in Fig. 5.

Only endpoints of the shadow region that are not equal to 0 or 1 correspond to centers of circles V_{trh} and V_{thr}. Therefore, no Voronoi circles exist in cases (1) and (2) of Fig. 5. Only V_{thr} (resp. V_{trh}) exists (assuming that $\mathbf{C_t}$, $\mathbf{C_r}$, $\mathbf{C_h}$ correspond to top site, bottom site and the shaded site of Fig. 5) in cases (3) and (4). Finally, both Voronoi circles exist in cases (5) and (6). Cases (2), (4) and (6) of Fig. 5 imply that $\mathrm{DFB}(\mathbf{C_t},\mathbf{C_r},\mathbf{C_h})<0$. This is so, because insertion of a third site which is in conflict with the CCW bitangent line, invalidates a part of the bisector that goes to infinity, as shown in the figure.

Table 1 (columns **C**–**H**) shows all possible combinations for the shadow region, and the evaluation of DFB. These combinations imply in turn the form the Voronoi diagram of three sites, as shown in Fig. 6.

Lemma 8. *The possible combinations for the shadow region of three sites are those presented in Table 1.*

Proof. Consider all permutations of 3 sites $\mathbf{C_t}$, $\mathbf{C_r}$, $\mathbf{C_h}$. There are two permutation groups generated by cyclic permutations of $(\mathbf{C_tC_rC_h})$ and $(\mathbf{C_tC_hC_r})$. Due to symmetry, it suffices to consider only one of the two, since the bisector of two sites is independent of the order in which the sites are considered. However for the oriented bisector (mapped to (0,1)), points at infinity are reversed. Therefore we have the following equivalences:

$S_{tr}(\mathbf{C_h})=\emptyset \Leftrightarrow S_{rt}(\mathbf{C_h})=\emptyset$, $\quad S_{tr}(\mathbf{C_h})=(0,1)\Leftrightarrow S_{rt}(\mathbf{C_h})=(0,1)$,
$S_{tr}(\mathbf{C_h})=(0,a)\Leftrightarrow S_{rt}(\mathbf{C_h})=(b,1)$, $\quad S_{tr}(\mathbf{C_h})=(b,1)\Leftrightarrow S_{rt}(\mathbf{C_h})=(0,a)$,
$S_{tr}(\mathbf{C_h})=(a,b)\Leftrightarrow S_{rt}(\mathbf{C_h})=(a',b')$,
$S_{tr}(\mathbf{C_h})=(0,a)\cup(b,1)\Leftrightarrow S_{rt}(\mathbf{C_h})=(0,a')\cup(b',1)$.

Next, we show that a form of $S_{tr}(\mathbf{C_h})$ implies a form of $S_{rh}(\mathbf{C_t})$ and a form of $S_{ht}(\mathbf{C_r})$. For each permutation in the cyclic group, consider the shadow region of the third element with respect to the first two. That is for the cyclic group $(\mathbf{C_tC_rC_h})$ we consider $(S_{tr}(\mathbf{C_h}),S_{rh}(\mathbf{C_t}),S_{ht}(\mathbf{C_r}))$, cf. columns **C**,**E**,**G** of Tab. 1. Due to the cyclic symmetry, all cyclic permutations of the shadow region forms are possible. Therefore, if we prove that $(S_{tr}(\mathbf{C_h}),S_{rh}(\mathbf{C_t}),S_{ht}(\mathbf{C_r}))$ can be $(\emptyset,\emptyset,(0,1))$, then it can also be $(\emptyset,(0,1),\emptyset)$ or $((0,1),\emptyset,\emptyset)$ (cf. rows 1 and 2 of the table).

Rows 1 and 2 of Table 1. Let $S_{tr}(\mathbf{C_h})=\emptyset$. Then, $S_{rh}(\mathbf{C_t})$ and $S_{ht}(\mathbf{C_r})$ are equal to either $\emptyset$ or $(0,1)$. If this was not the case, there would exist an endpoint of some shadow region, not equal to 0 or 1, which would be associated with a (tritangent) Voronoi circle. Such circles are always associated with points of the shadow region inside (0,1). This is a contradiction, since $S_{tr}(\mathbf{C_h})=\emptyset$. Moreover, it has to be that $S_{rh}(\mathbf{C_t})\neq S_{ht}(\mathbf{C_r})$. Because if $S_{rh}(\mathbf{C_t})=S_{ht}(\mathbf{C_r})=\emptyset$, then the Voronoi diagram of the three sites consists of three non-intersecting infinite edges (bisectors). This subdivides the plane into four disjoint cells associated with three sites, which forces a Voronoi cell of some site to consist of two disjoint cells according to the pigeonhole principle. This

is a contradiction, because every Voronoi cell is connected in abstract Voronoi diagrams. Similarly, if $S_{rh}(\mathbf{C_t}) = S_{ht}(\mathbf{C_r}) = (0, 1)$, then the Voronoi diagram consists of a single bisector that subdivides the plane in two cells. This is a contradiction, since each site should be associated with a different Voronoi cell. Therefore, the only possible cases of the shadow region are those presented in rows 1 and 2 of Tab. 1.

Rows 3 and 4 of Table 1. Let $S_{tr}(\mathbf{C_h}) = (0, a)$. Then, the endpoint a is associated with a Voronoi vertex which is the center of a Voronoi circle. Since there exists only one Voronoi vertex, the form of $S_{rh}(\mathbf{C_t})$, and $S_{ht}(\mathbf{C_r})$, is either $(0, a)$ or $(b, 1)$; as it has to contain exactly one endpoint which is not at infinity. It follows that $S_{rh}(\mathbf{C_t}) = S_{ht}(\mathbf{C_r}) = (0, a)$, because $S_{tr}(\mathbf{C_h}) = (0, a) \Rightarrow$ the part $(0, a)$ of the oriented bisector of $\mathbf{C_t}$ and $\mathbf{C_r}$ is destroyed $\Rightarrow$ part $(a, 1)$ remains $\Leftrightarrow$ part $(0, a)$ of the oriented bisector of $\mathbf{C_r}$ and $\mathbf{C_t}$ remains $\Rightarrow$ part $(0, a)$ of the oriented bisector of $\mathbf{C_r}$ and $\mathbf{C_h}$ is destroyed $\Rightarrow S_{rh}(\mathbf{C_t}) = (0, a)$. This argument justifies line 3 of Tab. 1 and by symmetry, line 4.

Rows 5 and 6 of Table 1. Let $S_{tr}(\mathbf{C_h}) = (a, b)$. Since there exist exactly two Voronoi vertices, the form of $S_{rh}(\mathbf{C_t})$ and $S_{ht}(\mathbf{C_r})$ is either (a, b) or $(0, a) \cup (b, 1)$; as it has to contain two endpoints not at infinity. If $S_{rh}(\mathbf{C_t}) = S_{ht}(\mathbf{C_r}) = (a, b)$, then the Voronoi diagram consists of 3 infinite bisectors of two connected components, whose finite endpoints coincide pairwise (If this was not the case, then we would have more than 2 Voronoi vertices, which a contradiction). This is also not possible as those 3 infinite bisectors would subdivide the plane into 5 cells, while the Voronoi diagram of the three sites has only 3 cells. In the same spirit, if $S_{rh}(\mathbf{C_t}) \neq S_{ht}(\mathbf{C_r})$ and $S_{rh}(\mathbf{C_t}) = (0, a) \cup (b, 1)$, then the Voronoi diagram consists of 2 infinite bisectors of two connected components, the finite endpoints of which coincide pairwise, and 1 finite bisector connecting the endpoints of the other two. Such a Voronoi diagram contains 4 cells which is again a contradiction. Therefore, $S_{rh}(\mathbf{C_t}) = S_{ht}(\mathbf{C_r}) = (0, a) \cup (b, 1)$ that yields a Voronoi diagram with 3 cells. This justifies lines 5 and 6 of Tab. 1 and concludes the proof. □

The total number of cases in Tab. 1 is 8. Is suffices to evaluate DFB on the cyclic permutations of three sites to determine the form of the shadow region, as well as to compute the number of Voronoi circles (i.e. the number of vertices of the Voronoi diagram of three sites), cf. columns (**A**,**B**) of Tab. 1. Up to this point, we have silently assumed that all three sites contribute to the Voronoi diagram (i.e. there are no hidden sites). In Sec. 4.4, we present an approach to handle hidden sites, as well.

Lemma 9. *Vertex conflict can be decided by looking at the ordering of the tangency points of Voronoi circles V_{trs}, V_{trh} and V_{thr} on $C(t)$.*

Proof. It follows from def. 7 and the fact that Voronoi circles are associated with the endpoints of the shadow region. □

	A $\exists V_{trh}$	**B** $\exists V_{thr}$	**C** $S_{tr}(\mathbf{C_h})$	**D** trh	**E** $S_{rh}(\mathbf{C_t})$	**F** rht	**G** $S_{ht}(\mathbf{C_r})$	**H** htr
1	0	0	$\emptyset$	0 0	$\emptyset$ $(0,1)$	0 1	$(0,1)$ $\emptyset$	1 0
2	0	0	$(0,1)$	1	$\emptyset$	0	$\emptyset$	0
3	0	1	$(0,a)$	0	$(0,a)$	0	$(0,a)$	0
4	1	0	$(b,1)$	1	$(b,1)$	1	$(b,1)$	1
5	1	1	(a,b)	0	$(0,a)\cup(b,1)$	1	$(0,a)\cup(b,1)$	1
6	1	1	$(0,a)\cup(b,1)$	1 1	$(0,a)\cup(b,1)$ (a,b)	1 0	(a,b) $(0,a)\cup(b,1)$	0 1

Table 1: Shadow region computation. trh stands for $\mathrm{DFB}(\mathbf{C_t},\mathbf{C_r},\mathbf{C_h})<0$ (resp. rht and htr)

To conclude, EdgeConflictType determines the type of the conflict-region, by computing the type of intersection of each edge of the Voronoi diagram (represented by an interval $[a,b]$) with the shadow region. This type of intersection is called *edge conflict type*. Like shadow region, the edge conflict type may have one out of six possible forms: $\emptyset$, (a',b'), $[a,b')$, $(a',b]$, $[\alpha,b')\cup(a',b]$ and $[a,b]$.

4.4. Existence

At some parts of the algorithm, it is required to know whether the CCW Voronoi circle V_{trs} exists or not, for example to determine the type of shadow region, in order to decide how to update the diagram upon insertion of a new site. This is a generalization of the Existence predicate (Emiris and Karavelas, 2006) to pseudo-circles.

There is a straightforward way to determine if V_{trs} exists. Solve the algebraic system and look for V_{trs}. If it is not found, then we may conclude that V_{trs} does not exist. However, this is non-practical and we have already shown that we can decide whether a Voronoi circle exists or not, using only the DFB predicate. Here we show that we can also determine its type (i.e. external or internal).

Clearly, when sites do not intersect, there cannot exist an internal Voronoi circle. When sites intersect in pseudo-circles, we first have to determine whether a site is hidden by one or more other sites. This is performed by a process called *medial axis location*, described in (Karavelas and Yvinec, 2003b). We mention it here for completeness, along with some issues concerning the implementation. Let $M(\mathbf{C_t})$ denote the medial axis of site $\mathbf{C_t}$. We say that a point p of $M(\mathbf{C_t})$ is *covered* by some site $\mathbf{C_i}$ if the maximal disk in $\mathbf{C_t}$ centered at p is included in the interior $\mathbf{C_i}^\circ$. If all points $M(\mathbf{C_t})$ are covered, we say that site $\mathbf{C_t}$ is *hidden* and all sites $\mathbf{C_i}$ that have been covering points of $M(\mathbf{C_t})$ form a covering set of $\mathbf{C_t}$. Due to the fact that each point p of $M(\mathbf{C_t})$ is associated with at least two footpoints on the boundary, $C(t)$, and the fact that the medial axis is a tree (for simple closed shapes), we can start from a leaf vertex of the medial axis and represent the covered part by an arc on the $C(t)$. Note that medial axis location becomes trivial when a site is fully contained in another one, as we can immediately deduce that it is hidden. Fig. 6 (case 2d) shows an example

of three intersecting ellipses where one of them is hidden by the other two and therefore it does not contribute to the Voronoi diagram.

In the case of ellipses, the medial axis is part of the major axis, more precisely the segment joining the centers of the circles of maximum curvature (foci of the ellipse). Due to the pseudo-circles intersection, the covered part of the medial axis always contains an endpoint (leaf vertex) (Karavelas and Yvinec, 2003a, Thm. 4), and can therefore be represented by an arc. We use a variation of system (2) to represent the footpoints of points p that correspond to the covered part of the medial axis. In the case of ellipses, they are represented as algebraic numbers of degree 16.

The following lemma combines Tab. 1 and location of medial axis, to correctly determine the existence and type (internal or external) of Voronoi circles in the case of hidden sites.

Lemma 10. *Given sites* $\mathbf{C_t}$, $\mathbf{C_r}$, $\mathbf{C_s}$, *let* κ *be the number of conflicts of* DFB, *when evaluated at triplets (*$\mathbf{C_t}$, $\mathbf{C_r}$, $\mathbf{C_s}$*), (*$\mathbf{C_r}$, $\mathbf{C_s}$, $\mathbf{C_t}$*) and (*$\mathbf{C_s}$, $\mathbf{C_t}$, $\mathbf{C_r}$*). Then (i) If* $\kappa < 2$ *then* V_{trs} *does not exist. (ii) If* $\kappa = 2$ *then: (a) If* $\mathbf{C_t} \cap \mathbf{C_r} \cap \mathbf{C_s} = \emptyset$, *then* V_{trs} *exists and is external. (b) Otherwise,* V_{trs} *exists iff there are no hidden sites and it is internal iff there exists a* $[t, r, s]$ *sequence of arcs on the boundary of the intersection (i.e. a CCW sequence of arcs belonging to* $\mathbf{C_t}$, $\mathbf{C_r}$ *and* $\mathbf{C_s}$ *respectively.) (iii) If* $\kappa = 3$ *then* V_{trs} *exists; it is internal iff* $\mathbf{C_t} \cap \mathbf{C_r} \cap \mathbf{C_s} \neq \emptyset$.

Proof. (i) $\kappa < 2$. From Tab. 1 we see that $\kappa < 2$ holds only for cases (1), (2) and (3), where the shadow region is of the form $\emptyset$, $(0, 1)$ or $(0, a)$ respectively. When the shadow region is $\emptyset$ or $(0, 1)$, it is obvious that no Voronoi circle exists. We see that the remaining case $(0, a)$ is associated with a circle of opposite orientation (V_{tsr}). Therefore V_{trs} does not exist. See also Fig. 6, case (0), where sites $\mathbf{C_t}$,$\mathbf{C_r}$,$\mathbf{C_s}$ correspond to labels 1,2,3 resp., as well as case (1), where sites correspond to labels 1,3,2 resp.

(ii) $\kappa = 2$. Without loss of generality we may assume that $\mathrm{DFB}(\mathbf{C_t}, \mathbf{C_r}, \mathbf{C_s}) \geq 0$ and $\mathrm{DFB}(\mathbf{C_s}, \mathbf{C_t}, \mathbf{C_r}) < 0$ (cf. case (5) of Tab. 1, we don't have to consider case (6) since that one is a cyclic permutation of (5)). (a) Case $\mathbf{C_t} \cap \mathbf{C_r} \cap \mathbf{C_s} = \emptyset$. It is obvious that in this case, V_{trs} cannot be internal. It follows (cf. Fig. 6 case 2a) that V_{trs} exists and is external (in fact, the other Voronoi circle, V_{tsr} exists as well, since the shadow region is of the form (a, b), therefore these endpoints have to be associated with a Voronoi circle each.) (b) Case $\mathbf{C_t} \cap \mathbf{C_r} \cap \mathbf{C_s} \neq \emptyset$. Given the possible cases of the Voronoi diagram of three sites (Fig. 6 2b, 2c, 2d) we have that if C_s is hidden (figure case 2d) then V_{trs} does not exist; otherwise, if $\mathbf{C_s}$ is not hidden, then V_{trs} exists, since the shadow region is of the form (a, b). V_{trs} is internal in this case iff there exists a $[t, r, s]$ sequence of arcs (true for figure case 2b, 2c when sites $\mathbf{C_t}$,$\mathbf{C_r}$,$\mathbf{C_s}$ correspond to labels 1,2,3 resp., but false for figure case 2b, when sites correspond to labels 2,1,3 resp.)

(iii) $\kappa = 3$. From Tab. 1 we see that $\kappa = 2$ holds only for case (4), where the shadow region is of the form $(b, 1)$ and V_{trs} always exists in this case, since it is associated with endpoint b of the shadow region. Now if $\mathbf{C_t} \cap \mathbf{C_r} \cap \mathbf{C_s} = \emptyset$, V_{trs}

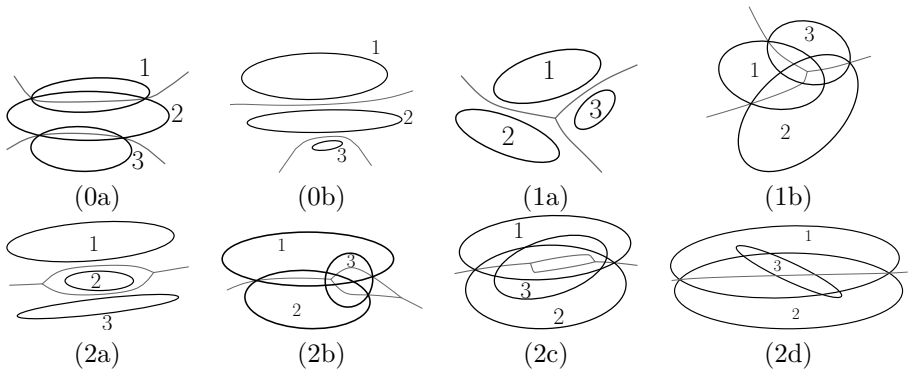

Figure 6: Voronoi diagram of 3 ellipses. The vertices of the diagram are centers of internally or externally tritangent circles. (0) No CCW or CW circle; (1a) Only external CCW circle (corresponding to order 1-2-3); (1b) Only internal CCW circle; (2a) Both circles exist and are external; (2b) External CW circle and internal CCW circle; (2c) Both circles exist and are internal; (2d) No Voronoi circles exist, because ellipse "3" is hidden

should necessarily be external, since it exists. Otherwise, in the case of common intersection, V_{trs} should be internal, because we know that only one Voronoi circle exists (since it is associated with endpoint b of the shadow region) and moreover, pairwise bisectors of $\mathbf{C_t}$,$\mathbf{C_r}$ and $\mathbf{C_r}$,$\mathbf{C_s}$ intersect and their point of intersection (center of Voronoi circle) lies in the intersection of all three sites (cf. case (1b) of Fig. 6). □

5. Implementation & experiments

This section describes our efficient, complete, and exact implementation for ellipses in the plane, possibly intersecting as pseudo-circles (Fig. 1), based on the algorithms we presented in the previous sections. We exploit the efficiency of our implementation on various datasets.

Our code is based on the existing CGAL Apollonius package for the combinatorial part of the algorithm. CGAL follows the generic programming paradigm and so the main issue is to implement the predicates for ellipses by generalizing the ones for circular sites. Some minor modifications in the combinatorial part of the algorithm are required in order to handle hidden ellipses correctly. A circle is hidden iff it is contained in some other circle. However, for more general convex objects, such as ellipses, a site can be hidden without necessarily being fully contained in another one (cf. Sec. 4.4 and Fig. 6, 2d).

Our implementation requires and extensive set of algebraic and arithmetic tools, including interval arithmetic, algebraic number representation, multivariate polynomial representation, and multivariate resultant computation. The most demanding predicate is InCircle and in the sequel we elaborate on the main design and implementation choices that we incorporate for its realization.

To answer quickly *non-degenerate* cases of InCircle we adopt the subdivision-based algorithm of (Emiris and Tzoumas, 2008). If a Voronoi circle exists, the algorithm will converge to it quadratically (although, for arbitrary intersecting smooth curves, when we are interested in an internal Voronoi circle, the convergence of the subdivision-based algorithm might not be quadratic). We implement this algorithm using interval arithmetic of multi-precision floating point numbers that CGAL provides through the libraries MPFI and MPFR[5]. We no longer rely on ALIAS or MATHEMAGIX[6] as in Emiris et al. (2009). Using this subdivision-based algorithm, we avoid solving the underlying polynomial system exactly using costly algebraic techniques e.g., resultant computations, and performing the tests of Sec. 4.2. The subdivision step integrates the these tests and provides a quick and certified evaluation InCircle. At the heart of the subdivision method we have implemented a univariate interval Newton solver, that handles polynomials with interval coefficients. This allows us to solve quickly multivariate equations by plugging in interval approximations for each variable.

However, when MPFR precision is not sufficient (i.e. when there exists a *degenerate* Voronoi circle, tangent to 4 sites), we fall back to the exact algebraic method (cf. Thm. 3). Because, even though the subdivision algorithm can approximate the tangency points that correspond to the Voronoi circle up to any precision, degenerate cases cannot be decided by merely a subdivision algorithm (in the case of equal algebraic numbers, the enclosing intervals will always intersect which makes it impossible to separate the numbers). The resultant provides us the true separation bound (for more details we refer the reader to (Emiris and Tzoumas, 2008)) and therefore a stopping criterion. This is due to the fact that the (slower) resultant computation provides a polynomial $R(t)$ while the subdivision method converges to some root of R without knowledge of R. In the presence of degeneracies, where equal algebraic numbers are to be expected, knowledge of the polynomials provides a means to determine such equality; the subdivision algorithm by itself is insufficient.

Such a combined approach can be seen from a more general perspective. We perform "difficult" algebraic operations using interval arithmetic as a filter. When degeneracies or near-degeneracies occur, this leads to uncertain sign evaluations, that prevent us from deciding the corresponding predicates. In this case, resultant computations provide us with the necessary information to decide. They determine the exact number of bits we need to decide the predicate. Almost always, we have to increase the precision by a constant number of bits, that is orders of magnitudes less than the number of bits predicted by the theoretical separation bound.

Finally, we have implemented a visualization algorithm for the bisector of two ellipses. The implicit equation of the bisector has total degree of 28 in the Cartesian space, that makes it impossible to handle numerically. To overcome

[5] http://www.mpfr.org/

[6] http://www.mathemagix.org/

this obstacle we trace the equivalent implicit curve in parametric space, where the degree is 12. This approach works quite well in practice. The bisectors in Figure 1 were traced and sketched using this visualization routine.

Overall, the design of our software package is generic and distinguishes the geometric from the algebraic computations. This is an important feature, that allows us to connect and interact with various algebraic software libraries as long as certain interface requirements are satisfied. Our final goal is to submit our implementation as a CGAL package, making it an official part of the library in one of the coming future releases.

5.1. Benchmarks

Predicate evaluation. In this section, we present various experimental results. All runtimes are obtained on a Core 2 Duo (T7200) 2.0GHz machine with 2GB of RAM, running Debian GNU/Linux 6.0 (Squeeze). We have measured the performance of SoB, DFB and InCircle with varying bitsize. Fig. 7 (top) corresponds to ellipses with randomly perturbed coefficients (axes, rotation and center of ellipses) by adding/subtracting 10^{-e}, where e varies. This forces the polynomials computed during each predicate evaluation to have coefficients of very large bitsize, since these coefficients depend on the input coefficients (axes, rotation, center) which are now perturbed. All runtimes appear to grow subquadratically (almost linearly in non-degenerate cases) in e. This is expected since SoB, DFB have constant arithmetic complexity (in terms of the bitsize of the polynomials that occur in the computations). Looking closely at Fig. 7, we see that non-degenerate and degenerate cases for SoB and DFB are decided extremely fast; it takes about 1 sec for input coefficients of 1600 bits (DFB is decided somewhat faster than the other ones). However, runtimes for near-degenerate cases scale more rapidly due to the fact that near-degenerate cases trigger a big number of subdivision steps for comparing algebraic numbers. For example, we decide in about 20 sec when the coefficients have 1600 bits. We should point out that our Algebraic Kernel benefits from constant arithmetic complexity algorithms for computing with algebraic numbers of degree up to 4, as they are implemented in MATHEMAGIX. Unfortunately such algorithms are yet to be found for higher degree numbers.

The performance of the first two predicates in near-degenerate cases is equivalent to that of InCircle in both non-degenerate and near-degenerate cases. This comes as no surprise, because the implementation of InCircle always uses a subdivision-based scheme to quickly approximate the solution of system (2).

We have measured the time needed for the subdivision algorithm to reach a precision of 2^{-b}, using MPFR floats, in Fig. 8 top (top and right axes). The x axis shows the number $b/1000$, i.e., the precision achieved in Kbits. Since the algorithm exhibits quadratic convergence, it computes τ bits in $\mathcal{O}(\log \tau)$ iterations. Standard floating point precision (53-bits) is achieved in about 30 msec (and 7 iterations, due to the quadratic convergence), and 1 sec suffices for almost 6000 bits of precision (14 iterations), whereas a 50-Kbit approximation takes about 40 sec (17 iterations). This shows that while one can achieve enormous precision, the theoretical separation bound of several million bits (Emiris et al.,

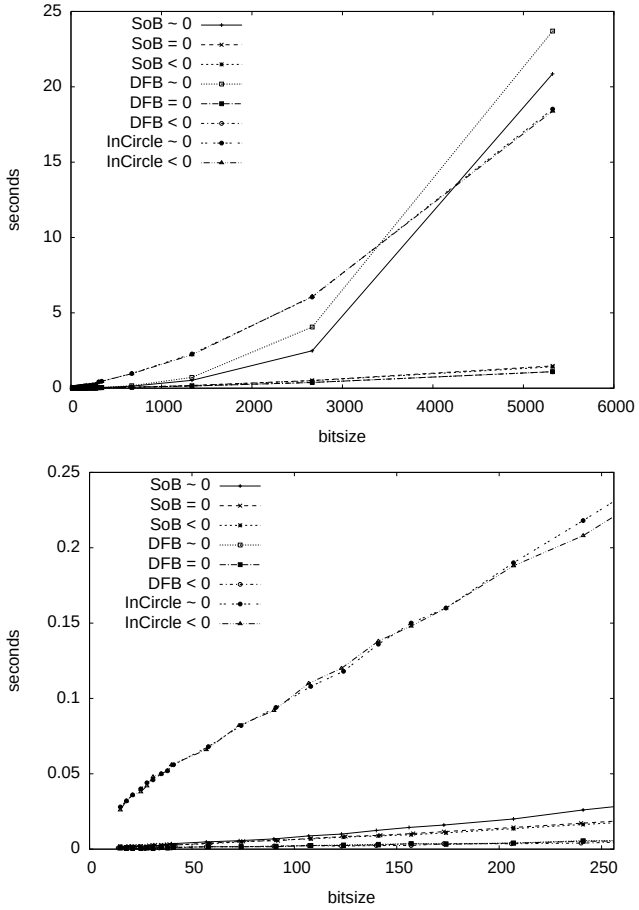

Figure 7: Top: Predicate performance, as bitsize b of the perturbed ellipses' coefficients grows; Bottom: Magnified portion of the same graph

2008) cannot be attained efficiently, hence the usefulness and the importance of resultant-based methods. Therefore, in case of degeneracies, the runtime of InCircle is dominated by the resultant computation, shown in Fig. 8 top (bottom and left axes). We observe that the resultant computation, even with 10-bit input coefficients, takes about 4 seconds to compute, i.e., 100 times slower than the subdivision algorithm using standard floating point precision of 2^{-53}. However, the algorithm is guaranteed to terminate in the case of degeneracies, while the subdivision scheme will run forever (trying to achieve the separation bound of several million bits, computing with numbers of huge precision). In short, both methods have to be combined for an exact, robust and efficient implementation.

The overall time for the construction of the Voronoi diagram (and the struc-

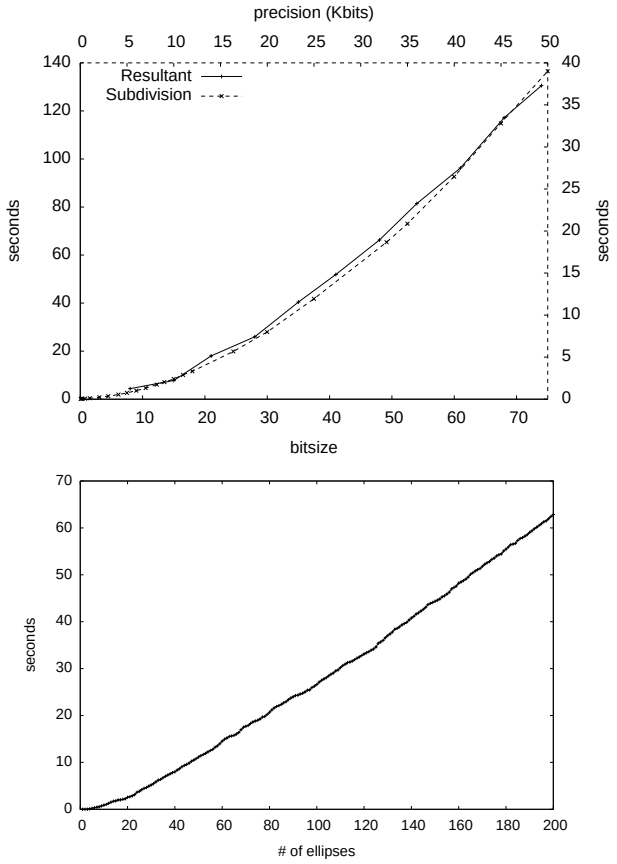

Figure 8: Top: Performance of resultant (bottom-left axes) and subdivision (top-right axes); Bottom: Benchmarking insertion of up to 200 ellipses

ture representing its dual graph) is shown in Fig. 8 bottom. It takes, for instance, 63 sec to compute the exact Voronoi diagram of 200 non-intersecting ellipses. More importantly, it is about linear in the number of sites. The runtime for intersecting ellipses is affected heavily by the number of times that location of medial axis has to be performed, as a resultant is being computed in that case. This fact suggests that we could benefit again from a subdivision scheme, as for InCircle. Fig. 9 shows the Voronoi diagram of 15 intersecting ellipses with 10-bit coefficients, as computed by our implementation. The runtimes vary between 1 and 5.5 sec. Fig. 6 was also generated by our implementation and shows a few more instances of intersecting ellipses, i.e., the possible cases of the Voronoi diagram of 3 ellipses.

Currently, we are trying to parallelize several parts of the implementation to take advantage of modern multi-core technologies in order to boost performance. We have already adjusted the drawing routine of the bisectors. Since

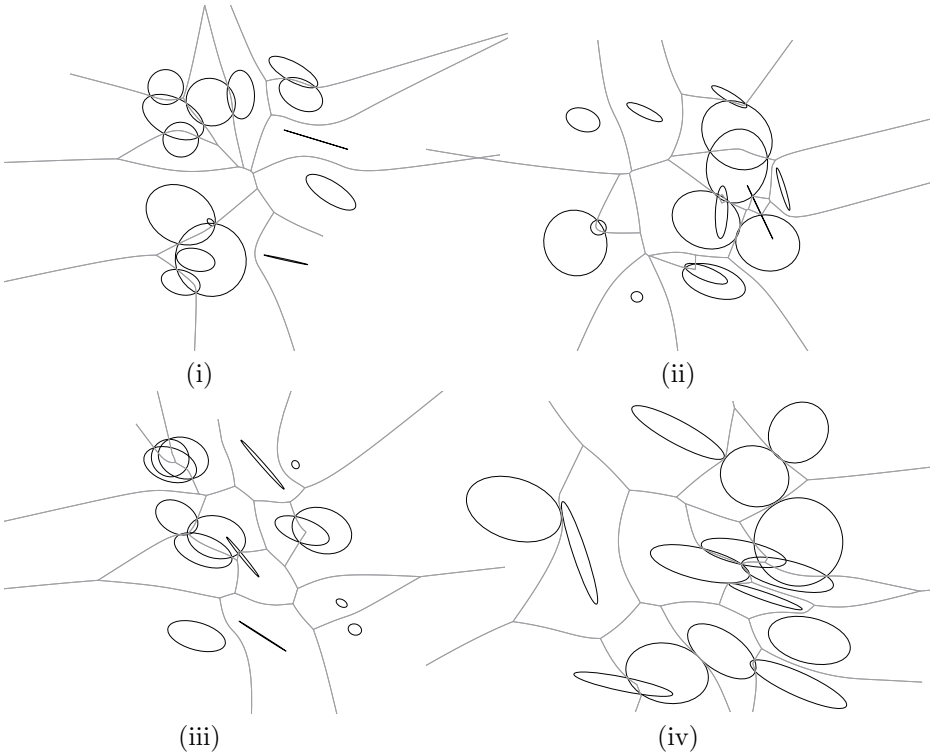

Figure 9: Runtimes in seconds with several instances of intersecting ellipses: (i) 1.36, (ii) 5.51, (iii) 1.87, (iv) 2.46

the Delaunay graph is stored in memory, the drawing routine has to visualize the bisectors one by one. This is currently performed in a parallel loop using OpenMP[7]. Parallelization may require modifications to CGAL itself. For example, the evaluation step in the resultant computation by interpolation which is performed by the Algebraic Kernel, can be done in parallel. We also perform some other optimizations like caching the outcome of expensive predicates, in case they are reused. For instance, during the construction of the diagram, InCircle quickly approximates Voronoi circles (i.e. bisector endpoints). These are stored and are readily available for the drawing routine as starting points to begin tracing.

Comparing with point approximations. An alternative way to solve problems with curved sites is to approximate them by simpler objects such as polygons or even sets of points. However, a good approximation may require a large number of input sites. Each ellipse is approximated by a constant number of k points taken uniformly on its boundary (cf. Fig. 10 left and middle). These points

[7] http://www.openmp.org/

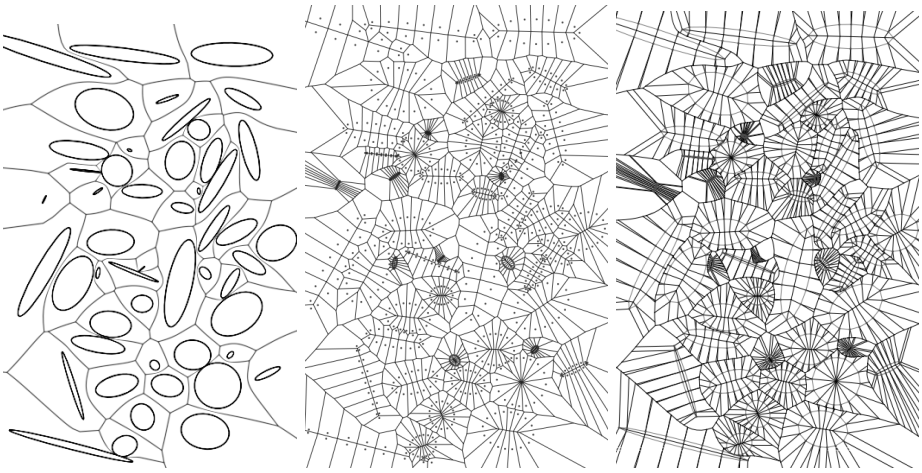

Figure 10: Left: Voronoi diagram of 50 ellipses; Middle: Voronoi diagram of 800 points, when each ellipse is approximated by 16 points; Right: Voronoi diagram of 50 16-gons approximating each ellipse (800 segments in total)

have rational coordinates, as they are obtained using (1). Using GNU rational arithmetic[8] (Gmpq in CGAL), we compare against 4 variations of the incremental algorithm of CGAL for the Delaunay triangulation: (i) without filtering,[9] (ii) with filtering, (iii) with filtering and improved nearest neighbor location: points are inserted in CCW order for each ellipse, so the Delaunay face of the lastly inserted point is given as a hint for the insertion routine, (iv) with filtering and spacial sorting.

Our own implementation for ellipses uses Gmpq but no filtering, except for InCircle that uses the subdivision-based method. This means that the whole implementation could be accelerated further if filtering is implemented in all predicates.

Fig. 11 left presents results concerning 50 ellipses, with k varying from 120 to 1250. We see that the Delaunay graph computation of 50 ellipses is faster for variations (i),(ii),(iii),(iv) of the Delaunay triangulation of points for $k \geq 120$, $k \geq 160$, $k \geq 320$ and $k \geq 1250$ respectively. The corresponding Delaunay triangulation have 600, 800, 1600 and 6250 vertices respectively. There are Delaunay edges corresponding to pairs of points on the same ellipse (with dual Voronoi edges in the ellipses' interior). These should be discarded which induces an extra overhead not measured. This is also true for all but one Delaunay edge between neighboring ellipses.

[8] http://gmplib.org/

[9] Filtering is a technique where the predicates are answered using double arithmetic, falling back to slower exact arithmetic only when the filter fails to produce an answer. This implies that there is some mechanism of determining that the outcome of the filter is correct or not.

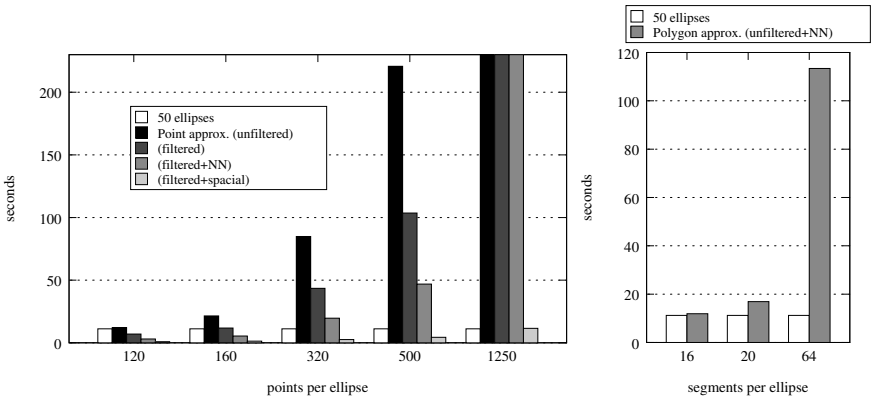

Figure 11: Performance of Voronoi diagram computation of 50 ellipses against: Point approximations with increasing number of points per ellipse (left), vertical axis clipped for readability (clipped values from left to right: 1369, 622 and 282.8); 50 polygons with increasing number of edges (right)

curve	d	**time**	$d(\mathsf{Res})$	**bound**
Beans	4	530.00	2632	6120
Conics	2	7.90	184	512
B-splines	3	9.20	404	2275
B-splines	2	0.65	93	512

Table 2: Resultant degree for various curves

Comparing with polygonal approximations. We compare against the CGAL package for the segment Voronoi diagram (more precisely, the segment Delaunay graph, Karavelas (2004).) The cost of an insertion is roughly $\mathcal{O}(\log^2 k)$, where k is the number of already inserted sites. We replaced each ellipse by a polygon (cf. Fig. 10 left and right). Care has been taken to perform smartly the nearest neighbor queries. In particular, since each segment is added in CCW order around each ellipse, a probable nearest neighbor is the lastly inserted segment. This is given as a hint to the insertion routine. In these benchmarks, the segment Delaunay implementation uses Gmpq arithmetic and no filtering.

Fig. 11 right shows the time required to construct the Voronoi diagram of 50 ellipses (white bar) and that of 50 polygons approximating each ellipse with 16, 20 and 64 edges (gray bar). Interestingly, the Delaunay graph of polygons is slower with > 15 segments per ellipse. As the number of edges per ellipse increases, the squared-logarithmic cost per insertion becomes non-negligible.

Experiments with general parametric curves. While our C++ implementation covers only ellipses for now, we have applied the proposed approach for the resultant computation on various types of curves using MAPLE. Some preliminary results are summarized in Tab. 2. The first column shows the type of curve, the second its degree, the third the time in sec, the fourth the degree of the resultant and the last column shows the (non-tight) bound of our general formula

from Cor. 5. First, we took the bean curve and applied simple affine transformations yielding very small (5-bit) coefficients. We computed the resultant of such triplets. The long runtimes indicate that working with high-degree curves requires very efficient implementations in order to be practical. We additionally considered the case of 3 random *conics* with small (10-bit) coefficients. In this case where $d = 2$, we have a tight bound of 184, while the general formula yields 512. We also tested the resultant computation on polynomial branches of degree 2 and 3 (*B-splines*), with very small (5-bit) coefficients. The runtime is less than 1 sec for $d = 2$ and less than 10 sec when $d = 3$. These experiments indicate that an efficient exact implementation may still be possible for larger bitsizes, and we can benefit from the increased flexibility that piecewise functions offer (with a trade-off between the total number of non-linear objects and the complexity of the algebraic operations).

6. Future work

Our first priority is to submit this work as a CGAL package. In the long run, we plan to adjust our algorithms for the predicates so as to compute the Voronoi diagram (or the convex hull) of convex sites with richer parametric representation, such as piecewise smooth parametric curves like NURBS or splines. The main issue is to identify efficiently the curve, or piece, which matters, and to apply the current predicates. This can be achieved quickly by numerical certified methods, such as our subdivision algorithm for the case of InCircle. Moreover, existing algorithms regarding Voronoi diagrams of arbitrary (possibly non-convex) non-intersecting curves (Hanniel et al. (2005); Seong et al. (2008)) may also benefit from our robust and exact approach, as their main computation is in fact the computation of a junction point of the Voronoi diagram, that is a Voronoi vertex, which can be expressed by the solution of eq. (2). Finally, the closely related problem of medial-axis computation can also be attacked by our techniques, as bifurcation points of the medial axis correspond (in general) to solutions of system (2). There are already some preliminary results in this direction (Tzoumas, 2011).

It would be also interesting to study the similarity of the classical Voronoi diagram of certain classes of sites (i.e. ellipses) with the anisotropic Voronoi diagram of points, as the latter could provide an alternative means of approximation.

Another extension would be to remove the pseudo-circles constraint by applying the idea of Karavelas and Yvinec (2003b): Given two sites $\mathbf{C_t}$ and $\mathbf{C_r}$, then the Voronoi diagram in free space depends only on the arcs appearing on the boundary of the union of the sites (Fig. 12). This way we can handle arbitrarily intersecting objects by treating them as degenerate pseudo-circles (that are externally tangent in one point). Note that while the Voronoi diagram is computed in free space (i.e. the complement of the objects' union), it is sufficient for a wide range of applications.

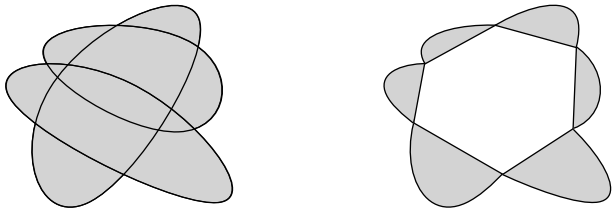

Figure 12: Handling arbitrarily intersecting convex sites

7. Acknowledgments

I. Emiris is partially supported by Marie-Curie Network "SAGA", FP7 contract PITN-GA-2008-214584. E. Tsigaridas is partially supported by the EXACTA grant of the National Science Foundation of China (NSFC 60911130369) and the French National Research Agency (ANR-09-BLAN-0371-01), GeoLMI (ANR 2011 BS03 011 06), HPAC (ANR ANR-11-BS02-013) and an FP7 Marie Curie Career Integration Grant. Most of the work of G. Tzoumas was performed during his postdoc at INRIA Nancy and a smaller part during his PhD at the National and Kapodistrian University of Athens.

References

Aichholzer, O., Aigner, W., Aurenhammer, F., Hackl, T., Jüttler, B., Rabl, M., 2009. Medial axis computation for planar free-form shapes. Computer-Aided Design 41, 339 – 349. Voronoi Diagrams and their Applications.

Alt, H., Cheong, O., Vigneron, A., 2005. The Voronoi diagram of curved objects. Discr. and Comput. Geometry 34, 439–453.

Anton, F., 2004. Voronoi diagrams of semi-algebraic sets. Ph.D. thesis. The University of British Columbia.

CGAL, . CGAL: Computational Geometry Algorithms Library. http://www.cgal.org.

Cox, D., Little, J., O'Shea, D., 2005. Using Algebraic Geometry. Number 185 in GTM, Springer, New York. 2nd edition.

Elber, G., Kim, M.S., 1998. Bisector curves of planar rational curves. Comput. Aided Des. 30, 1089–1096.

Elber, G., Kim, M.S., Heo, H.S., 2001. The convex hull of rational plane curves. Graph. Models 63, 151–162.

Emiris, I., Tsigaridas, E., Tzoumas, G., 2009. Exact Delaunay graph of smooth convex pseudo-circles: general predicates, and implementation for ellipses, in: SPM '09: 2009 SIAM/ACM Joint Conf. on Geom. & Phys. Model., San Francisco, CA, USA. pp. 211–222.

Emiris, I.Z., Karavelas, M.I., 2006. The predicates of the Apollonius diagram: algorithmic analysis and implementation. Comp. Geom.: Theory & Appl. 33, 18–57. Spec. Issue Robust Geom. Algorithms & Implement.

Emiris, I.Z., Tsigaridas, E.P., Tzoumas, G.M., 2008. Predicates for the exact Voronoi diagram of ellipses under the Euclidean metric. Intern. J. Computational Geometry & Applications 18, 567–597. Spec. Issue on SoCG'06.

Emiris, I.Z., Tzoumas, G.M., 2008. Exact and efficient evaluation of the InCircle predicate for parametric ellipses and smooth convex objects. Comput. Aided Des. 40, 691–700.

Habert, L., 2005. Computing bitangents for ellipses, in: Proc. 17th Canad. Conf. Comp. Geom., pp. 294–297.

Hanniel, I., Muthuganapathy, R., Elber, G., Kim, M.S., 2005. Precise Voronoi cell extraction of free-form rational planar closed curves, in: Proc. ACM Symp. Solid Phys. Modeling, Cambridge, MA. pp. 51–59.

Held, M., 2001. Vroni: An engineering approach to the reliable and efficient computation of Voronoi diagrams of points and line segments. Comput. Geom. Theory Appl. 18, 95–123.

Held, M., Huber, S., 2009. Topology-oriented incremental computation of Voronoi diagrams of circular arcs and straight-line segments. Comput. Aided Des. 41, 327–338.

Karavelas, M.I., 2004. A robust and efficient implementation for the segment Voronoi diagram, in: Proc. Int. Symp. Voronoi Diagrams, pp. 51–62.

Karavelas, M.I., Yvinec, M., 2003a. The Voronoi Diagram of Convex Objects in the Plane. Research Report RR-5023. INRIA.

Karavelas, M.I., Yvinec, M., 2003b. Voronoi diagram of convex objects in the plane, in: Proc. Europ. Symp. Algorithms, Springer. pp. 337–348.

Kim, D.S., Kim, D., Sugihara, K., 2001. Voronoi diagram of a circle set from Voronoi diagram of a point set: II. Geometry. Comp. Aid. Geom. Des. 18, 563–585.

Klein, R., Mehlhorn, K., Meiser, S., 1993. Randomised incremental construction of abstract Voronoi diagrams. Comput. Geom.: Theory & Appl. 3, 157–184.

Ogarko, V., Luding, S., 2010. Data structures and algorithms for contact detection in numerical simulation of discrete particle systems, in: World Congress Particle Technology 6, Nürnberg Messe GmbH, Nuremberg, Germany. CD-Proceedings.

Ramamurthy, R., Farouki, R., 1999a. Voronoi diagram and medial axis algorithm for planar domains with curved boundaries - II: detailed algorithm description. J. Comput. Appl. Math. 102, 253–277.

Ramamurthy, R., Farouki, R., 1999b. Voronoi diagram and medial axis algorithm for planar domains with curved boundaries I. theoretical foundations. J. Comput. Appl. Math. 102, 119–141.

Ramanathan, M., Gurumoorthy, B., 2003. Constructing medial axis transform of planar domains with curved boundaries. Comp.-Aided Design 35, 619–632.

Seong, J.K., Cohen, E., Elber, G., 2008. Voronoi diagram computations for planar NURBS curves, in: Proc. ACM Symp. Solid & Phys. modeling, NY. pp. 67–77.

Tzoumas, G., 2009. Computational geometry for curved objects. Voronoi diagrams in the plane. Ph.D. thesis. National Kapodistrian University of Athens.

Tzoumas, G., 2011. Exact medial axis of quadratic NURBS curves, in: 27th Proc. Europ. Workshop Computat. Geometry, Morschach, Switzerland. pp. 91–94.